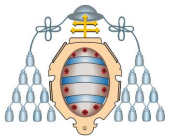


The Convolution of Signals

1 Goals

The aim of this laboratory session is to illustrate the practical aspects of the convolution operation, which have been described in the lectures. The goal of this session is twofold. First, a Matlab function will be implemented so it performs a graphical animation of the convolution of two signals. Second, the qualitative meaning of the convolution between an arbitrary signal and a Gaussian function, which happens in many physical problems, will be studied.

2 Convolution of aperiodic signals

In this first section, the goal is to implement the convolution of two functions defined in Matlab. For this purpose, some new tools will be introduced and a pseudocode describing a possible implementation will be described.

2.1 Tools

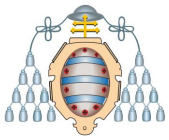
Some new tools will be firstly introduced to accomplish the implementation of the convolution of analytical signals. First, the convolution of continuous-variable signals requires evaluating an integral, whose solution is not always analytical and, therefore, it must be evaluated by means of numerical techniques. Although an in-depth study of the numerical evaluation of integrals is out of the scope of this course, the student must know that any function $f(x)$ evaluated in a vector `f` at the points in the vector `x`, can be estimated by means of the command `trapz(x,f)`.

It is important to note that the numerical evaluation of integrals usually entails two approximations: i) the information about the signal $f(x)$ is only available at a discrete set of points `x`; ii) the integrand (i.e., the signal) of the improper integral can only be evaluated over a finite interval $[a, b]$, which typically corresponds to the points `a=x(1)` and `b=x(end)`. These two approximations regarding signal discretization and interval truncation result in *numerical errors*. Nevertheless, these errors can be acceptable as long as enough points are used to model the function and the interval covers most of the signal energy.

$\int_{-\infty}^{\infty} f(x)dx \cong \int_{a}^{b} f(x)dx \Rightarrow$

```
x = linspace(a,b,N); %N is the number of points
integral = trapz(x,f);
```

The second tool is the use of *anonymous functions*. From a practical point of view, these functions work as any function such as the ones defined on conventional Matlab M files (i.e., extension *.m) . Thus, the function can be evaluated at a vector of points `x` as any other



function `f(x)`. However, the main advantage is that this kind of functions can be passed as arguments to other functions. Next, an example of a definition of a piecewise function by means of an anonymous function is shown:

$$f(x) = \begin{cases} 1 & x < -1 \\ x^2 & -1 \leq x \leq 1 \\ 2 & x > 1 \end{cases} \Rightarrow$$

```
1 f = @(x) 1      * (x<-1)           + ...
2          x.^2 .* (x>=-1 & x<=1)   + ...
3          2      * (x>1);
```

Please, note that the logical operators (`<`, `>`, `&`, etc.) enable us to easily define the piecewise function.

Finally, the third tool, which will be relevant for this session, is composed of the basic commands to create animations in Matlab. As it happens when printing the output of a C (or similar languages) program to the console, the output is not automatically refreshed and it is sometimes required to force an update. Although this fact is usually not observed when drawing a single plot, it is required when creating animations. For this purpose, the following command, which plays a role similar to *flush* in C, is used:

```
>> drawnow;
```

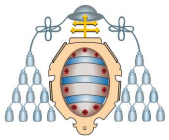
The interested reader is referred to the help of this command to find its advanced options.

2.2 Pseudocode for the convolution

The pseudocode to perform a convolution is shown in Algorithm 1. Take into account that each line of this pseudocode can be translated into just one or two lines in Matlab, which should provide guidance with respect to the correct (in terms of compactness) implementation of the convolution.

In addition to the file for the convolution function, it is recommended to write a second file comprising several definitions and the call to your new convolution function. In this second file, the values corresponding to the sampling of the variables x and x' as well as the definition of the *anonymous functions* corresponding to $f(x)$ and $g(x)$.

! In this session, it is recommended to set the integration interval as $x' \in [-10, 10]$ and the observation interval as $x \in [-3, 3]$. Regarding the sampling, the steps $\Delta x' = 0.01$ and $\Delta x = 0.1$ are recommended for the integration x' and observation variables x, respectively.



input : Handle to anonymous function `f`
input : Handle to anonymous function `g`
input : Vector `x` containing the values x where $c(x)$ will be evaluated
input : Vector `x_` containing the values x' where the convolution integral will be performed
output: Values of $c(x)$ evaluated at `x`

`f_evaluated` ← Values of `f` at `x_`;
Initialize `c` with zeros `%c = zeros(1, length(x));`
foreach *element `x0` of the vector `x`* **do**
  `g_evaluated` ← Values of `g` at `x0-x_`;
  `c(x0)` ← Value of the integral of `f_evaluated.*g_evaluated` along `x_`;
  Draw $f(x')$ and $g(x_0 - x')$;
  Draw (the available data of) $c(x)$;
  `drawnow`;
end

Algorithm 1: Pseudocode for the convolution of continuous-variable aperiodic signals.

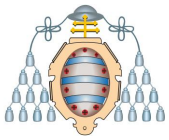
TASK

1. Implement a function in Matlab to compute the convolution between two functions as described in Algorithm 1.
2. Use the previous function to calculate the convolution of $f(x) = \Pi(\frac{x}{2})$ and $g(x) = e^{-x}u(x)$.
3. Compare the previous result with the analytical solution:

$$c(x) = \Pi(\frac{x}{2}) * \left[e^{-x}u(x)\right] = \begin{cases} 0 & x \leq -1 \\ 1 - e^{-x-1} & -1 \leq x \leq 1 \\ 2e^{-x}\sinh 1 & x \geq 1 \end{cases}$$

OPTIONAL

1. Use the previous function to numerically calculate and plot the signal $c(x) = f(x) * g(x)$ where $f(x) = x\Pi(\frac{x}{2})$ and $g(x) = e^{-x}$. The same values for x', x, $\Delta x'$ and Δx can be used.
2. Compare the result with the analytical solution $c(x) = 2e^{-(x+1)}$. Do you observe any difference? Why do the results not perfectly match?



OPTIONAL

1. Implement a function to perform the periodic convolution. Note that this convolution is defined almost identical to the previous one so you only need to change the integration interval.
2. Implement a function to perform the convolution of two discrete-variable aperiodic signals. Note that this definition is almost identical to the previous one so you only need to replace the integral (function `trapz`) by a summation (function `sum`)

3 Impact of the convolution

3.1 Impact on 1D signals

In order to better understand the convolution, let us focus on a single point x_0 at which the convolution is given by:

$$c(x_0) = \int_{-\infty}^{\infty} f(x')g(x_0 - x')dx'$$

the previous operation can be understood as the average of a signal $f(x')$ around x_0 weighted by the signal $g(-x')$, which is conveniently moved to x_0.

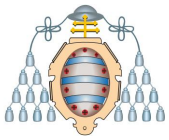
TASK

1. Compute the convolution of a pulse $f(x) = \Pi(\frac{x}{2})$ and a Gaussian function $g(x) = e^{-\frac{x^2}{2\sigma}}$ with $\sigma = 0.04$.
2. Represent, either on paper or with Matlab, the functions $f(x')$, $g(x_0 - x')$ and $f(x')g(x_0 - x')$ and reflect on the involved averaging.
3. Compare the shape of the pulse before and after the convolution. What differences do you observe?

3.2 Impact on 2D signals

The previous effect can be observed in many usual signals such as pictures. Although the in-depth study of 2D signals is out of the scope of this course, the impact on an image of applying a convolution with a 2D Gaussian function is illustrated to qualitatively understand the effect of this operation, which is very usual in many graphics editors for different purposes (e.g., Photoshop uses it under the command "Gaussian blur")

```
1 I = imread('cameraman.tif'); %Loads an image from Matlab images
2 figure; imshow(I); title('Original image'); %Shows the original image
3 PSF = fspecial('gaussian',7,10);
4 I_blurred = conv2(double(I),double(PSF)); %Performs a 2D convolution ...
      values are converted from unit8 to double
5 figure; imshow(uint8(I_blurred)); title('Blurred image'); %Shows the image ...
      after convolution (requires converting to unit8)
```



TASK Run the previous code (the *Image Processing Toolbox* is required) and find the qualitative differences between the before and after images.